# *Pole (array)*

## Motivace

- Častá úloha – práce s větším množstvím dat stejného typu
  - Př.: průměrná teplota za týden a odchylka od průměru v jednotlivých dnech

```
Console.Write("Zadej T pro 1.den: ");
double t1 = Double.Parse(Console.ReadLine());
Console.Write("Zadej T pro 2.den: ");
double t2 = Double.Parse(Console.ReadLine());
// atd.
Console.Write("Zadej T pro 7.den: ");
double t7 = Double.Parse(Console.ReadLine());

double tPrumer = (t1 + t2 + t3 + t4 + t5 + t6 + t7) / 7;
Console.Write("\nPrumerna teplota je {0}", tPrumer);

Console.Write("Odchylka T pro 1.den: {0}", t1 - tPrumer);
Console.Write("Odchylka T pro 2.den: {0}", t2 - tPrumer);
// atd.
Console.Write("Odchylka T pro 7.den: {0}", t7 - tPrumer);
```

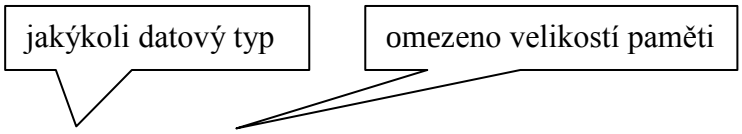
  - nelze řešit cyklem s postupným výpočtem průměru
  - nevýhody:
    - algoritmus nevykazuje znaky hromadnosti (teploty za měsíc, rok???)
    - obtížná manipulace s daty (jak na hledání extrémů?)

## Základní pojmy

- pole = strukturovaný datový typ, složen z většího množství položek (prvky pole) – (item)
- pole = homogenní datová struktura („Pole prvků typu …“ – „array of …“) – všechny prvky pole stejný datový typ
- index prvku (item index) – pozice prvku od začátku pole

## Vytvoření pole

- pole = proměnná
- deklarace:




```
typ[] JmenoPole = new typ[pocetPrvku];
```

  - Příklady:

```
const int POCET_PRVKU = 7;
double[] teplota = new double[POCET_PRVKU];
string[] zaznamy = new string[10000];
```

## Práce s polem

### Po prvcích

- typicky v cyklu `for`

```
const int POCET_PRVKU = 7;
double[] teplota = new double[POCET_PRVKU];
double tPrumer = 0;
// nacteni prvku z klavesnice
for (int i = 0; i < teplota.Length; i++)
{
      Console.Write("Zadej T pro {0}.den: ", i + 1);
      teplota[i] = double.Parse(Console.ReadLine());
      tPrumer += teplota[i];
}

// vypocet a vypis prumeru
tPrumer /= teplota.Length;
Console.WriteLine("Prumerna teplota je {0}", tPrumer);
// vypocet a vypis odchylek
for (int i = 0; i < teplota.Length; i++)
{
      Console.WriteLine("Odchylka T pro {0}.den: {1}",
            i + 1, teplota[i] - tPrumer);
}
```

Vlastnost;počet prvků

Index

```
C:\WINDOWS\system32\cmd.exe
Zadej T pro 1.den: 12
Zadej T pro 2.den: -32
Zadej T pro 3.den: 0.1254
Zadej T pro 4.den: 12.45
Zadej T pro 5.den: -0.002
Zadej T pro 6.den: 10
Zadej T pro 7.den: 8.9

Prumerna teplota je 1.639057

Odchylka T pro 1.den: 10.360943
Odchylka T pro 2.den: -33.639057
Odchylka T pro 3.den: -1.513657
Odchylka T pro 4.den: 10.810943
Odchylka T pro 5.den: -1.641057
Odchylka T pro 6.den: 8.360943
Odchylka T pro 7.den: 7.260943
Pokračujte stisknutím libovolné klávesy...
```

### Poznámky

- výsledek výrazu pro index: celé číslo

```
teplota[3]
teplota[2*k+1]          // liché prvky
teplota[1.5]            // špatně
```

  - typicky proměnná typu `int` pojmenovaná `i`
- prvek pole = proměnná jednoduchého typu

```
double[] teplota = new double[7];
tPrumer += teplota[i];
dalsiPole[5] = 3*sin(teplota[2*i]);
```

- Meze pole → vždy od 0 do počet prvků – 1

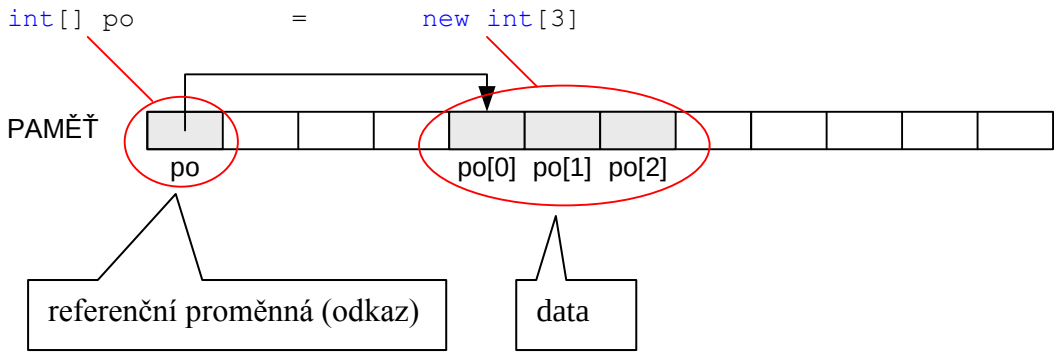
```
teplota[0]              // 1. prvek
teplota[3]              // 4. prvek
teplota[POCET_PRVKU-1]  // poslední prvek
teplota[POCET_PRVKU]    // nejčastější chyba
```

- Délka pole: Vlastnost jen pro čtení `Length` typu `int`

```
int pocetPrvku = teplota.Length;    // 7
```

## Pole a paměť

```
int[] po = new int[3];
```




- lze rozdělit deklaraci odkazu a „vznik“ pole:

odkaz nikam nesměřuje (obsah proměnné: `null`)

```
int[] po;          // prazdny odkaz
po[2] = 10;        // chyba, pole není vytvořeno
po = new int[12]; // muzeme pouzivat
po[2] = 10;        // OK
```

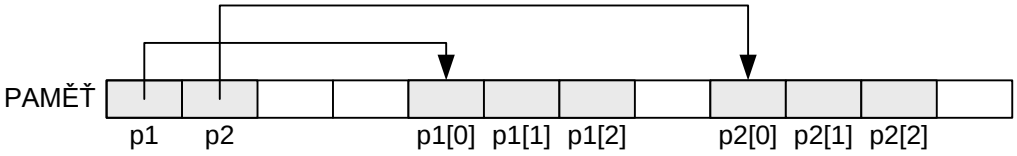
- velikost pole → i „za běhu programu“

```
Console.Write("Zadej delku pole: ");
int delka = Int32.Parse(Console.ReadLine());
double[] pole = new double[delka];
```

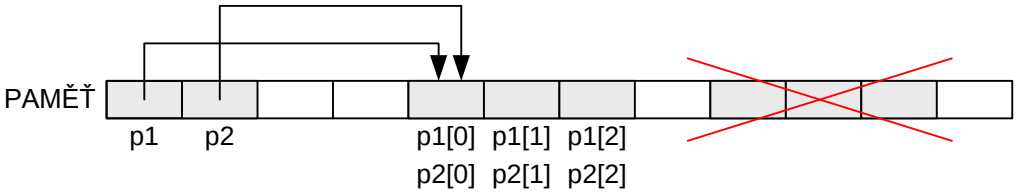
### Práce s referenční proměnnou

```
int[] p1 = new int[3];
int[] p2 = new int[3];  // 1
p2 = p1;                // 2
```

- po 1




- po 2 → `p2` odkazuje na stejná data jako `p1`




  - jakákoli změna v `p2` se projeví se i v `p1` (a naopak)
  - k původním datům v poli `p2` se již nelze dostat!!!

### Inicializace pole

- Nové pole → prvky hodnotu: `0`, `0.0`, `false`, …
- inicializace

```
int[] p2 = new int[3] { 1, 2, 3 };
int[] p2 = { -87, 3, 24, 589 };     // délku doplní překladač
```

## *Vícerozměrná pole*

- = pole s více indexy
  - 2 indexy = matice, nejčastější případ
  - 3 indexy = „3D“ pole („kvádr“)
- Základní principy = 1D pole
- C# dva typy:
  - Pravoúhlé (2D – tvar obdélníku)
  - Nesymetrické – každý řádek → jiný počet sloupců

### Pravoúhlá

- pouze 2D
- deklarace:

```
typ[,] jmenoPole = new typ[radku, sloupcu];
```

  - např. `int[,] pole = new int[2, 3];`

- Inicializace v deklaraci

```
int[,] pole =
{
      {11, 12, 13},
      {21, 22, 23}
}
```

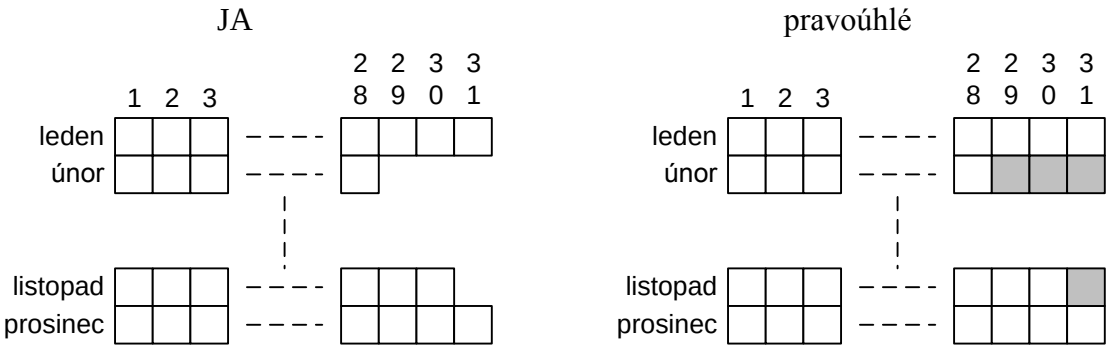
„2D pole = 1D pole, jehož prvky jsou 1D pole“

- Zjištění rozměrů:
  - Vlastnost `Rank` → počet dimenzí pole
  - Vlastnost `Length` → počet všech prvků
  - Metoda `int GetLength(int dimenze)` → vrací počet prvků v dimenzi `dimenze`
- Práce s vícerozměrným polem → nejčastěji vnořené cykly `for`
  - Příklad: Součet dvou matic $\boldsymbol{C} = \boldsymbol{A}+\boldsymbol{B}$

```
const int RADKU = 2;
const int SLOUPCU = 3;
int[,] A = new int[RADKU, SLOUPCU];
int[,] B = new int[RADKU, SLOUPCU];
int[,] C = new int[RADKU, SLOUPCU];
for (int i = 0; i < A.GetLength(0); i++) // pres radky
{
      for (int j = 0; j < A.GetLength(1); j++)
      // v radku pres sloupce
      {
            C[i, j] = A[i, j] + B[i, j];
      }
}
```

## Nesymetrická

- (Jagged Array)
- každý řádek → jiný počet sloupců
- důvody: úspora místa, rychlost
- příklad uložení informace o průměrné teplotě po dnech, měsících v jednom roce



# *Pole a metody*

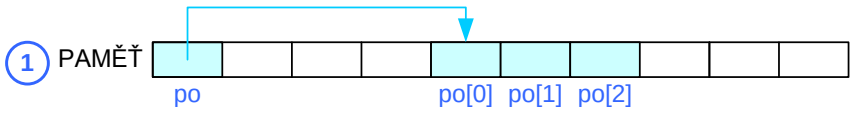
## Pole jako parametr

- 1D i vícerozměrná pole stejné
- Příklad: metoda, která vytiskne pole na obrazovku

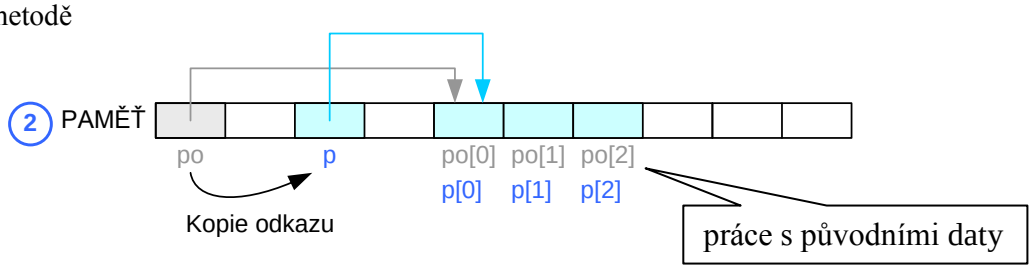
```
static void Main(string[] args)
{
      int[] po = new int[5];
      po[1] = 45;
      TiskPole(po);
}
static void TiskPole(int[] p)
{
      for (int i = 0; i < p.Length; i++)
      {
            Console.WriteLine(p[i]);
      }
}
```

- do metody se předává (kopíruje) pouze referenční proměnná, nikoli kompletní data!!! = volání odkazem:
  - rychlé
  - prvky pole lze uvnitř metody změnit

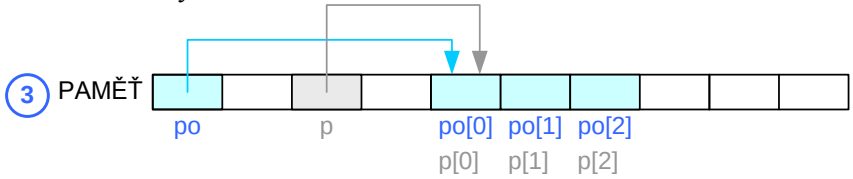
Před voláním metody




V metodě




Po skončení metody



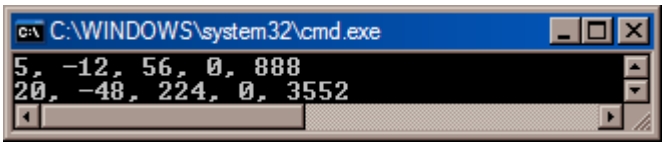

  - na počtu prvků nezáleží

- Příklad: Metodu, která vynásobí pole skalárem

```
static void Main(string[] args)
{
      int[] po = new int[5] { 5, -12, 56, 0, 888 };
      PoleKratSkalar(po, 4);
}
```

není nutné `ref` či `out`

```
static void PoleKratSkalar(int[] p, int skalar)
{
      for (int i = 0; i < p.Length; i++)
      {
            p[i] *= skalar;
      }
}
```




## Pole jako návratová hodnota metody

- vrací se pouze odkaz, s daty se nic neděje → rychlé
- Příklad: Metoda, která sečte dva vektory

```
static int[] SectiVektory(int[] v1, int[] v2)
{
      if (v1.Length != v2.Length)
            return null;
      int[] vektor = new int[v1.Length];
      for (int i = 0; i < vektor.Length; i++)
      {
            vektor[i] = v1[i] + v2[i];
      }
      return vektor;
}
static void Main(string[] args)
{
      int[] pole1 = { 1, 2, 3 };
      int[] pole2 = { 1, 2, 3 };
      int[] soucetPoli = SectiVektory(pole1, pole2);
}
```

výsledek výrazu = odkaz na data